EE524 Exercise 7 a & b

NOTE: All Ex7 exercises will use the Lena (lena_512x512_32bit.png) image provided.
(except Bonus items)

EX7a

In VS17 create a new empty OpenCL project on your local drive (under C:\ not local user directory!)
Copy the Ex7_starter_hostapp.cpp from classweb into your new VS17 EX7 project directory.
In Solution Explorer, Add Existing to add the Ex7_starter_hostapp.cpp to your project Source Files
Dthe default host.cpp
Rename the default device.cl to ex7_kernels.cl
Set VS17 Build configuration to x64
Check project properties to ensure using SDK Version = 10.x and Platform Toolset = Visual Studio 2017 (v141)
Ensure the starter host app code builds (it wont run yet without proper kernel *.cl's and relevant host code mods).

Starting from your GaussBlur filter from L6/EX6,
create a copy of the filter kernel code, and rename the kernel function to SobelGrad().
Implement the two 3x3 (horizontal & vertical) Sobel gradient stencils as 3x3 __constant float arrays at file scope
Modify the core stencil convolution loop to compute both the Gx and Gy convolved filter sums
Add implementations to compute
(1) true magnitude of gradient vector value
(2) fast approximation of gradient vector value
(refer to L7 slides)
note: you'll need to use float-capable built-in math functions: fabs, fmin, etc... (see OpenCL C Spec, sec 6.13.2)

Use host app to apply gradient stencils, save output images, and observe effect differences.

Create another copy of the GaussBlur filter code, rename the filter to Laplacian().
Implement two 3x3 Laplacian stencils as 3x3 __constant float arrays at file scope
(1) basic Laplacian stencil (with diagonal terms)
(2) composite Laplacian stencil (with diagonal terms)

Use host app to apply both Laplacian filters, save output images, and observe effect differences

BONUS: Implement a simple point-based gray-level transformation.
Implement contrast enhancement using the power-law transformation to increase image dynamic range.

- use a value of gamma < 1.
- try starting with gamma = 0.5 and c = 1, and experiment from there.

EX7b

Add a new item to your EX7 project: Source File, C++ file
Name: simple_devNQ.cl
Copy the simple device-queue kernel from the L7 slides.
Configure the host app to support device-side command-queue, and launch an NDRange of global_size = {2,2,0} (see L7 slides).
Compile, build, and run simple device-queue kernel from commandline.
Observe printf outputs to console and ensure parent and child kernels execute as expected.

Based on the simple device-queue kernel and the multi-filter master kernel pseudocode in L7 slides,
Create a master kernel to run the following sequence of child kernels
: sobel-->gaussblur-->laplacian-->rotate
Note: as in example pseudocode, all child kernels will need to be in same *.cl file and precede (above) the parent kernel definition.

Use host app to enqueue the master kernel using work size of 1.
Incrementally modify your master kernel to run the following stages of filters
Save output result images at each step
(1) Sobel gradient only
(2) Sobel-->GaussBlur
(3) Sobel-->GaussBlur-->Laplacian
(4) Sobel-->GaussBlur-->Laplacian-->Rotate
Observe and compare the various stage results.

BONUS: replace final Rotate stage with the power-law contrast enhancement gray-level transform filter.

- Apply to the medical xray image samples.